SLAC Chat

SLAC's Unofficial Simple Chat Service

Use:

- Clone repository: https://github.com/Jacob-Le/Chat_App.git
- cd into Chat_App
- Start up the server with:
    - python server.py --host 127.0.0.1 --port 33002
    - Optionally start with "python server.py &" to run in background
    - The default port is 33002 and the host is 127.0.0.1
- Currently, only the Linux SLAC Chat executable is available.
    - Win32, Win36, and Mac OSX executables are WIP
    - The Linux executable can be found in the root folder, either use ./SLAC_Chat or double click the icon if you are using the Linux GUI
- Alternatively, you can use python to run:
    - python SLAC_Chat.py --host 127.0.0.1 --port 33002
    - Like server.py, the default port is 33002 and the host is 127.0.0.1
- Login
- Enjoy SLAC Chat!
- If you have any further questions, feel free to email me at emailboxofjacob@gmail.com

Building an executable:

- Requirements:
    - Linux/Ubuntu environment
    - Qt
    - Sip
    - PyQt5
    - pyinstaller
- CD into SLAC_Chat
- Run pyinstaller SLAC_Chat.py --onefile --clean --hidden-import PyQt5.sip
- The build executable binary will be under ./dist/SLAC_Chat

Documentation:

SLAC_Chat.py:

- Main python file where the Qt application loop runs. Use this to run SLAC Chat.

chat_window.py:

- Module that contains implementation for the main Chat Window
    - Features include:
        - Automatically connects to server upon startup
        - Chat text box + send button (pressing enter/return serves the same function as clicking send)
        - Combo box that allows user to choose a specific user to send messages to, or alternatively send to everyone in the server.
        - Auto-scrolling chat log
        - Confirmation prompt upon exiting the application
        - Connection status label that displays the client's current connection status
        - Menu options for notification muting, connection, disconnection, and quitting the app (using 'Quit SLAC Chat' is functionally the same as using the window's 'x' button)

Object/function	Description
ChatUI	UI code generated from QtDesigner, slightly edited.
ChatWindow	Subclasses ChatUI and QMainWindow, contains all chat window functionality
ChatWindow.prompt_login()	Starts running the client on a separate QThread, and prompts the user to register with a new username. Spawns a LoginDialog window.
ChatWindow.add_msg_item(str)	Adds messages to the chat log, alternating between white and grey for better visibility. Appends current time to the message.
ChatWindow.send_chat_message()	Sends a message taken from existing text from the chat box and sends it to the recipient specified in the

	ComboBox.
ChatWindow.got_msg(str)	pyqtSlot that listens for a msg_recieved signal from the client. The msg_recieved signal comes with the message read from the socket buffer. If notification mute is turned off, it will not play the notification sound.
ChatWindow.update_info()	pyqtSlot that listens for a update_widgets signal from the client. Updates the ComboBox with the roster of users that is sent by the server.
ChatWindow.update_conn(bool)	pyqtSlot that listens for a conn_status signal from the client. The conn_status signal comes with the connection status to the server in the form of a bool value. Updates the connectionStatusLabel and disables/enables the appropriate widgets.
ChatWindow.on_sendButton_clicked()	pyqtSlot connected to the sendButton. Calls ChatWindow.send_chat_message(). Functionally identical to pressing the return/enter key.
ChatWindow.toggle_mute()	pyqtSlot connected to actionMute. Toggles notification mute.
ChatWindow.action_connect()	pyqtSlot connected to the connect button in the menubar. Is only available when disconnected from the server. Spawns a ConnectDialog window, prompting the user to connect using a (Host, Port) address. (Attempts to connect to default if nothing is entered) Calls ChatWindow.prompt_login() afterwards.
ChatWindow.action_disconnect()	pyqtSlot connected to the disconnect button in the menubar. Much like actionQuit, prompts the user with a confirmation MessageBox window. Once ‘Yes’ is selected, the client thread will stop, wait until it has completed, and disconnect from the server.
ChatWindow.quit_button_event() and close_event(event)	Much like action_disconnect, prompts the user witha confirmation MessageBox window. Once ‘Yes’ is selected, the client thread will stop, wait until it has completed, and disconnect from the server. The QApplication will then quit.

client.py

- Module that contains client functionality, is able to connect to the server. Subclasses QObject such that it may run in the same loop as the rest of the application. Contains pyqtSignals that notify the Chat Window when messages are received.
- Features:
  - TCP connection to the server
  - Message processing/separating via a delimiter, currently set to ‘^’

Object/function	Description
Client	Client object that functions very close to the chat window
Client.run()	Loops while connected to the server. Processes incoming messages and disconnects gracefully if connection is lost.
Client.read_msg()	Helper function that reads from the socket buffer. Separates messages via a delimiter, currently set to ‘^’
Client.set_address(str, str)	Sets the address to which the client will attempt to connect to.
Client.connect()	Creates a new socket and attempts to connect to the stored address. Emits results via the conn_status signal.
Client.disconnect()	Sends “{QUIT}” message to the server and closes the socket. Emits results via the conn_status signal.
Client.register(str)	Sends “{REGISTER}” message to the server, along with a username.
Client.send_message(str, prefix=””)	Low level helper function that combines prefix and msg, converts to bytes and sends across the socket.

connect_dialog.py:

- Module that contains implementation for a ConnectDialog window that allows the user to specify a host:port combination for the client to connect to.
- Features:
    - Host:port vetting, ensures that valid hosts and ports are submitted.
    - Tool tips

Object/function	Description
ConnectUI	UI code generated from QtDesigner, slightly edited.
ConnectDialog	Subclasses ConnectUI and QDialog.
ConnectDialog.on_connectButton_clicked()	If host and port are left blank, will substitute in default host and port 127.0.0.1:33002. Checks if address and port are valid. Will reject and display error message via tooltip if not valid. Otherwise, will set host and port to input values and connect.
ConnectDialog.inform(str, error=bool)	Sets infoLabel to display string message. If error is True, display in red and return to default message over time. If error is False, display in black.

server.py:

- Functionally identical to the provided server.py, only difference is that the send_message function appends the delimiter “^” to the end of each message.

login_window.py:

- Module that contains functionality for a LoginDialog window.
- Features:
    - Username vetting: usernames cannot contain non-alphanumeric characters, the word “ALL” or be 0 characters in length, or be more than 24 characters in length.
    - Tooltips

Object/function	Description
LoginUI	UI code generated from QtDesigner, slightly edited.
LoginDialog	Subclasses LoginUI and QDialog.
ConnectDialog.on_loginButton_clicked()	Matches username against regex expressions to determine if it is valid. If it is, it will send the “{REGISTER}” message to the server.
ConnectDialog.inform(str, error=bool)	Sets infoLabel to display string message. If error is True, display in red and return to default message over time. If error is False, display in black.

Troubleshooting

- When unable to connect to the server before starting SLAC Chat:
  - Exit out of the login window.
  - Ensure server.py is running in a separate terminal or in the background
  - The Chat Window will be spawned. Go to Menu > Connect.
  - Enter server host IP address and port number.
  - Connect, and register with a username
- If you have any further questions, feel free to email me at emailboxofjacob@gmail.com

Known Bugs

- Currently no guards against users registering under the same name. Either the server will choose one user over the other to send messages to, or information will be mixed up.
- The notification beep may not play on all operating systems.

Future Features

- Encryption using AES and SHA-256.
- Bring up ConnectDialog window upon startup to allow users to connect to a different address
- Error handling system.
- Search through received messages using a search bar.
- Server connection reattempts.
- Ability to attempt to connect to the server from the login screen
- Ability to quit out of the application from the login window when not connected to the server and not be brought to the chat window.